Mejora de la Eficiencia en Sistemas de Atención al Ciudadano mediante IA: *Ayuda Inteligente*, *Detección Automática de Errores* y *Consultas Personalizadas*

Amparo Valenzuela Juan
Máster Universitario en Investigación en Inteligencia Artificial (AEPIA-UIMP)
Asociación Española para la Inteligencia Artificial (AEPIA)
Universidad Internacional Menéndez Pelayo (UIMP)
Curso académico 2024–2025
100007769@alumnos.uimp.es

Resumen* - Este trabajo presenta una arquitectura semántica e inteligente diseñada para mejorar la eficiencia de los sistemas de atención al ciudadano. La solución propuesta combina una ontología OWL implementada en el lenguaje con consultas SPARQL ejecutadas sobre una base de datos relacional Oracle mediante Ontop, permitiendo un acceso semántico a datos estructurados. Asimismo, se integra una interfaz de lenguaje natural basada en un modelo de lenguaje de gran tamaño (LLM), utilizando la API de OpenAI, que traduce automáticamente las preguntas formuladas por el usuario a consultas SPARQL. El sistema desarrollado ofrece tres mejoras clave: ayuda contextual inteligente, detección semántica de errores y consultas personalizadas sobre la base de datos. La solución ha sido evaluada comprobando la capacidad del sistema para interpretar preguntas en lenguaje natural, generar consultas SPARQL válidas y recuperar correctamente los datos reales desde la base de datos Oracle. La arquitectura propuesta es independiente del dominio y puede aplicarse a cualquier sistema de gestión que requiera acceso estructurado a datos, razonamiento semántico e interacción inteligente con el usuario.

atención al ciudadano, ayuda inteligente, detección semántica de errores, ontología, SPARQL

Abstract* - This paper presents a semantic and intelligent architecture designed to improve the efficiency of citizen service systems. The proposed solution combines an OWL ontology implemented in the language with SPARQL queries executed over a relational Oracle database through Ontop, enabling semantic access to structured data. Additionally, it integrates a natural language interface powered by a large language model (LLM) via the OpenAI API, which automatically translates user questions into SPARQL queries. The developed system offers three key improvements: intelligent contextual assistance, semantic error detection, and personalized database queries. The solution has been evaluated by testing the system's ability to interpret natural language questions, generate valid SPARQL queries, and accurately retrieve real data from the Oracle database. The proposed architecture is domain-independent and can be applied to any management system that requires structured data access, semantic reasoning, and intelligent user interaction.

citizen services, intelligent assistance, semantic error detection, ontology, SPARQL

Introducción

En la actualidad, muchas administraciones públicas emplean aplicaciones web para gestionar y difundir información dirigida a la ciudadanía a través de sus portales institucionales. Estos sistemas permiten consultar trámites, acceder a servicios, participar en convocatorias de empleo público, oposiciones o bolsas de trabajo, así como consultar información sobre la estructura organizativa o el contacto de unidades administrativas. Sin embargo, en numerosos casos, estas soluciones están basadas en arquitecturas tradicionales con bases de datos relacionales y sin integración de inteligencia artificial, lo que limita su eficiencia y capacidad de adaptación.

Uno de los principales retos detectados en este tipo de entornos es la ausencia de mecanismos inteligentes que asistan al usuario durante su interacción con el sistema. La ayuda disponible suele ser estática y genérica, sin capacidad para adaptarse al contexto específico de la acción que el usuario está realizando. Por otro lado, uno

de los problemas recurrentes es la aparición de errores en los datos tras actualizaciones masivas, migraciones o modificaciones no controladas. Estos errores pueden incluir duplicidades, inconsistencias entre campos o registros incompletos, y suelen detectarse solo una vez que han impactado en el funcionamiento del sistema. La falta de mecanismos automatizados para validar la coherencia y la integridad de los datos dificulta una gestión eficaz y aumenta el riesgo de errores persistentes en producción.

Asimismo, los usuarios responsables de gestionar el sistema se enfrentan a importantes limitaciones a la hora de consultar y extraer información personalizada. La aplicación no ofrece herramientas flexibles para generar listados específicos, por lo que deben recurrir al personal técnico para obtener consultas a medida que les permitan acceder a datos concretos. Esta dependencia ralentiza los procesos de análisis y publicación de información, especialmente cuando se requiere responder con agilidad a necesidades puntuales.

Este trabajo plantea una solución basada en Web Semántica e inteligencia artificial para mejorar la eficiencia de los sistemas de atención al ciudadano. La propuesta combina el diseño de una ontología del dominio con consultas SPARQL ejecutadas sobre bases de datos relacionales mediante Ontop, junto con una interfaz en lenguaje natural respaldada por modelos de lenguaje de gran tamaño (LLM) a través de la API de OpenAI. El sistema resultante permite ofrecer ayuda contextual inteligente, detectar errores semánticos en los datos y facilitar la generación de consultas personalizadas de forma accesible para el usuario.

La validación se ha realizado mediante casos de uso simulados que reproducen situaciones reales del ámbito de atención al ciudadano. En cada caso, se formularon preguntas en lenguaje natural y se comprobó si el sistema era capaz de interpretarlas correctamente, generar la consulta SPARQL adecuada y recuperar los datos esperados desde la base de datos relacional. Aunque el desarrollo parte de un entorno específico, la arquitectura propuesta es modular, escalable y puede aplicarse a otros sistemas de gestión que requieran interacción inteligente, validación semántica y explotación de datos estructurados.

La motivación de este proyecto surge de la experiencia directa de la autora en el mantenimiento funcional de un sistema real de atención al ciudadano en el ámbito de una administración pública. Esta vivencia permitió identificar de primera mano diversas carencias operativas y oportunidades de mejora relacionadas con la interacción del usuario, la gestión de errores y la explotación de datos. El objetivo principal es demostrar que la integración de tecnologías semánticas e inteligencia artificial permite optimizar el funcionamiento de estos sistemas sin necesidad de rediseñarlos desde cero. Además, se ha buscado comprobar empíricamente la eficiencia de la arquitectura propuesta, evaluando si la solución desarrollada cumple efectivamente con los objetivos planteados. De este modo, no solo se analiza su viabilidad técnica, sino también su aplicabilidad práctica en contextos reales de uso.

En conjunto, este trabajo pretende contribuir al avance de sistemas públicos más eficientes, accesibles y adaptables mediante la aplicación de tecnologías semánticas e inteligencia artificial.

Estado del arte

La aplicación de la Web Semántica y la inteligencia artificial en la gestión de información ha sido ampliamente investigada en los últimos años, especialmente en entornos donde la estructuración del conocimiento y la interacción inteligente con los datos son fundamentales. Este apartado revisa enfoques relevantes que sustentan la viabilidad técnica del modelo propuesto en este trabajo.

Firmenich [1] plantea una arquitectura que conecta chatbots con ontologías RDF/OWL para responder consultas conversacionales, mejorando la interacción entre usuarios y bases de datos semánticas. En una línea similar, Arteaga Meléndez [2] desarrolla un chatbot explicativo basado en ontologías, mientras que Ciancio [3] aplica técnicas de NLP para recuperar información histórica desde periódicos digitalizados. Estos enfoques confirman el potencial de la Web Semántica para generar respuestas contextualizadas y precisas.

La construcción automática de ontologías a partir de datos no estructurados ha sido abordada por Huaranga Junco [4] y por Gómez de Agüero Muñoz [5], quienes exploran cómo los LLMs como GPT-4 pueden extraer, refinar y gestionar modelos ontológicos, facilitando su escalabilidad. Tupayachi et al. [6] complementan esta línea con una propuesta orientada a sistemas de decisión urbana, donde los LLMs permiten estructurar grandes volúmenes de información y validarlos mediante inferencia semántica.

También se han explorado marcos basados en Web Semántica para mejorar la interoperabilidad y el análisis de datos. Paneque Romero [7] presenta un enfoque orientado a la estandarización y explotación semántica, mientras que Lagos Ortiz [8] demuestra cómo las ontologías pueden utilizarse para generar recomendaciones personalizadas en el ámbito agrícola.

En el contexto específico de la administración pública, Martínez [9] realiza una revisión sistemática sobre chatbots con inteligencia artificial y propone una arquitectura basada en NLP, aprendizaje automático y semántica. Celi-Párraga et al. [10] complementan esta visión con un análisis de técnicas clave de NLP, como el reconocimiento de entidades, transformers como BERT y GPT, y generación de respuestas contextuales.

Iniciativas como QALD (Question Answering over Linked Data) [11] han sido clave para evaluar de forma sistemática la capacidad de los sistemas de preguntas en lenguaje natural para acceder a datos estructurados mediante SPARQL, sirviendo de referencia para el diseño y la validación de soluciones similares.

Por último, Brito Karadjian [12] diseña un sistema que convierte peticiones en lenguaje natural en consultas SQL a través de un chatbot integrado con OpenAI, LangChain y PostgreSQL. Su objetivo es mejorar la eficiencia de usuarios no técnicos al interactuar con bases de datos estructuradas.

Estos estudios consolidan una base teórica robusta para la integración de ontologías, NLP y modelos LLM en sistemas de gestión de información, validando los pilares tecnológicos sobre los que se apoya la solución propuesta en este TFM.

Diseño e Implementación del sistema

Este apartado describe el diseño técnico y la implementación del sistema desarrollado, orientado a mejorar la eficiencia y la interacción en entornos de atención al ciudadano mediante tecnologías de Web Semántica e inteligencia artificial. La solución integra una ontología OWL del dominio, consultas SPARQL sobre base de datos relacional mediante Ontop, y una interfaz de usuario que permite realizar consultas en lenguaje natural.

La arquitectura ha sido concebida con un enfoque modular, escalable y reutilizable, permitiendo su aplicación en otros sistemas de gestión que requieran explotación semántica de datos y asistencia inteligente al usuario. A continuación, se detallan los componentes fundamentales del sistema, desde su estructura general hasta las funcionalidades clave implementadas.

I. Ontología del dominio y mapeo semántico

Con el objetivo de proporcionar una representación formal y explotable del conocimiento en el ámbito de atención al ciudadano, se ha diseñado una ontología específica en OWL que modela los principales conceptos, entidades y relaciones del dominio. Esta ontología constituye la base semántica del sistema, permitiendo interpretar los datos con significado y habilitando funcionalidades avanzadas como la ayuda contextual, la inferencia lógica o la generación de consultas personalizadas.

La ontología incluye clases como *Personas*, *Departamentos*, *TramiteServicio* (fases y subfases), *EmpleoPublico* (etapas), *AyudaCampo*, *Registros* y *Operacion*, entre otras, junto con sus propiedades de datos (por ejemplo, per*Nombre*, trs*Codigo*, fini*FechaIni*) y propiedades de objeto que describen las relaciones entre entidades (por ejemplo, *perPerteneceADep*, *trsPerteneceADep*). La modelización sigue principios de modularidad y reusabilidad, alineados con la metodología *NeOn* [13], y ha sido validada mediante el razonador ontológico *Pellet*, así como con la herramienta *OOPS!* (Ontology Pitfall Scanner!), para garantizar la coherencia lógica del modelo y detectar posibles errores estructurales o malas prácticas ontológicas.

Para conectar esta representación semántica con los datos reales del sistema, se ha utilizado *Ontop*, una plataforma de acceso a datos basada en ontologías (OBDA). A través de un archivo *.obda*, se han definido los mapeos que vinculan cada propiedad de la ontología con los atributos correspondientes de las tablas de una base de datos Oracle. Por ejemplo, la clase *Personas* se mapea con la tabla *AC_PERSONAS*, y sus propiedades con columnas como *PER_NOMBRE*, *PER_CODIGO* o *PER_UNIDAD*. De este modo, las consultas SPARQL escritas sobre la ontología se traducen automáticamente a instrucciones SQL y se ejecutan directamente sobre la base de datos relacional, sin necesidad de duplicar información ni alterar el modelo de datos original.

Esta separación entre la capa semántica y la capa física permite una evolución independiente de ambas, facilita el mantenimiento del sistema y promueve la reutilización del modelo en otros contextos administrativos con estructuras similares.

II. Arquitectura general del sistema

La solución se compone de cuatro capas principales: interfaz de usuario, capa de procesamiento semántico, motor de traducción SPARQL-SQL (Ontop) y base de datos Oracle. A través de un frontend desarrollado en React, el usuario puede acceder a funciones como la ayuda contextual o la realización de consultas personalizadas. Las preguntas formuladas en lenguaje natural son procesadas por el backend en Spring Boot, que se encarga de invocar a un modelo de lenguaje de gran tamaño (LLM) vía la API de OpenAI para generar consultas SPARQL a partir de la ontología definida.

Estas consultas se envían a Ontop, que actúa como middleware semántico, transformando dinámicamente las consultas SPARQL en instrucciones SQL sobre la base de datos relacional sin necesidad de replicar ni migrar los datos. Finalmente, los resultados se devuelven al usuario en un formato comprensible, con posibilidad de exportación.

Este enfoque permite desacoplar el modelo semántico de los datos físicos, proporcionando una solución flexible y no intrusiva.

La arquitectura completa se muestra en la Figura 1.

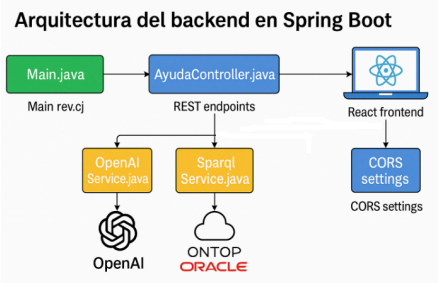


Figura 1. Arquitectura del backend en Spring Boot. El controlador *AyudaController.java* expone endpoints REST que interactúan con dos servicios principales: uno para la generación de consultas SPARQL mediante OpenAI (*OpenAIService.java*) y otro para su ejecución sobre Ontop (*SparqlService.java*). El frontend, desarrollado en *React*, se comunica con estos servicios a través de peticiones HTTP. La configuración *CORS* permite el intercambio de datos entre cliente y servidor sin restricciones. Fuente: elaboración propia.

III. Interfaz de lenguaje natural

Uno de los principales objetivos del sistema es permitir a los usuarios acceder a los datos sin necesidad de conocimientos técnicos, como SPARQL o SQL. Para ello, se ha integrado una interfaz de lenguaje natural que permite formular preguntas en lenguaje natural y obtener respuestas precisas extraídas directamente de la base de datos, a través de la capa semántica.

Esta funcionalidad se implementa en el backend, desarrollado en Spring Boot, mediante un componente que interactúa con la API de OpenAI. Cada vez que el usuario formula una consulta desde el frontend, el sistema envía dicha pregunta al servicio *OpenAIService.java*, que utiliza un modelo de lenguaje de gran tamaño (LLM) para generar dinámicamente una consulta SPARQL compatible con la ontología del dominio.

El prompt enviado a OpenAI ha sido cuidadosamente diseñado para guiar al modelo en la generación de consultas precisas, estructuradas y sintácticamente válidas. Para ello, se ha seguido una metodología específica que contempla la definición de un vocabulario autorizado, extraído de la ontología y de los mapeos OBDA del sistema, que delimita las clases y propiedades que pueden utilizarse. El prompt incluye también ejemplos representativos (*few-shot learning*) que ilustran cómo transformar preguntas frecuentes del dominio en consultas SPARQL correctas; por ejemplo, indicándole la instrucción *"Si el usuario menciona un término del diccionario, utiliza su campo técnico asociado en la consulta SPARQL"*. Además, se han incorporado instrucciones explícitas para asegurar que el modelo devuelva únicamente la consulta SPARQL, sin explicaciones adicionales, y utilice correctamente los tipos de datos (*xsd:string*, *xsd:integer*, etc.). Las consultas generadas se validan antes de su ejecución para garantizar su compatibilidad con el motor Ontop y con la estructura semántica del sistema. A lo largo del desarrollo, el diseño del prompt ha sido refinado de forma iterativa, incorporando mejoras progresivas tras detectar errores o comportamientos inconsistentes. Esta evolución ha dado lugar a distintas versiones del prompt, cuya efectividad se ha comparado mediante una evaluación específica sobre un subconjunto de preguntas seleccionadas. Los detalles de dicha comparativa se analizan en el apartado correspondiente de la evaluación. Además, en la **sección *Repositorio del proyecto*** se indica la ubicación exacta donde se encuentran disponibles los **prompts utilizados, los recursos técnicos, los mapeos OBDA, el código fuente del sistema, así como los resultados completos de las evaluaciones** realizadas durante el desarrollo del trabajo. El sistema valida la sintaxis de la consulta generada antes de ejecutarla. En caso de que la consulta SPARQL sea válida, se reenvía al componente *SparqlService.java*, que la ejecuta a través de Ontop sobre la base de datos Oracle, devolviendo los resultados al usuario en formato estructurado o exportable.

Este enfoque permite combinar la potencia expresiva de los modelos de lenguaje con el rigor formal de la Web Semántica, proporcionando una experiencia de usuario accesible, intuitiva y eficaz, especialmente útil en contextos donde los usuarios finales no disponen de conocimientos técnicos avanzados.

A considerar como consulta válida aquella que devuelve los datos esperados por el usuario, asumiendo por lo tanto que sintácticamente es correcta.

IV. Funcionalidades clave implementadas

El sistema desarrollado incorpora tres funcionalidades principales orientadas a mejorar la eficiencia, accesibilidad y fiabilidad en el entorno de atención al ciudadano. Estas funcionalidades se basan en la integración entre la ontología del dominio, el procesamiento en lenguaje natural y el acceso semántico a los datos reales almacenados en una base de datos relacional.

Ayuda contextual inteligente. Se ha implementado un botón de ayuda contextual que proporciona explicaciones dinámicas sobre los campos, secciones y operaciones disponibles en la interfaz. Esta ayuda se genera a partir de la ontología OWL, la cual contiene anotaciones semánticas y reglas que describen la estructura y funcionamiento de los elementos del sistema. El componente *AyudaController.java* del backend recibe el contexto actual del usuario (por ejemplo, la sección en la que se encuentra) y consulta Ontop mediante SPARQL para recuperar la información relevante, que luego se muestra en el frontend en tiempo real.

Detección semántica de errores. Para mejorar la calidad de los datos y prevenir problemas funcionales derivados de inconsistencias o registros incompletos, se ha incorporado una funcionalidad de detección de errores

semánticos. Se ha definido una clase *Error* en la ontología. Las anomalías se identifican mediante consultas SPARQL diseñadas para localizar patrones como duplicidades, relaciones incoherentes o valores ausentes en campos clave. Aunque la ejecución de estas consultas no es automática, podría lanzarse periódicamente o ante eventos críticos para monitorizar el estado semántico del sistema.

Consultas personalizadas en lenguaje natural. Los usuarios pueden realizar preguntas directamente en lenguaje natural, como "¿Qué personas están asignadas al Departamento Jurídico?" o "Muéstrame un listado que contenga nombre y descripción de trámites que tienen como destino a la Ciudadanía o Empresas". El sistema traduce automáticamente estas preguntas a consultas SPARQL mediante un modelo LLM de OpenAI, accede a la base de datos vía Ontop y devuelve los resultados de forma estructurada o exportable. Esta funcionalidad permite el acceso a la información de forma flexible e intuitiva, eliminando barreras técnicas para usuarios no especializados.

Evaluación

La evaluación del sistema se ha realizado con el objetivo de comprobar su eficacia, precisión y viabilidad a partir de una implementación funcional sobre datos simulados del dominio de atención al ciudadano.

Para cada una de las funcionalidades clave del sistema, se han definido conjuntos de pruebas específicas y se han aplicado métricas de evaluación tanto cuantitativas como cualitativas que permiten valorar el comportamiento del sistema desde diferentes perspectivas.

A continuación, se describen los criterios utilizados, los experimentos realizados y los resultados obtenidos para cada una de las tres mejoras implementadas: ayuda contextual inteligente, detección semántica de errores y consultas personalizadas en lenguaje natural.

I. Evaluación de la Ayuda Contextual Inteligente

Para evaluar el funcionamiento del sistema de ayuda inteligente contextual, se ha elaborado un **conjunto de 91 preguntas en lenguaje natural**, formuladas por el evaluador como lo haría un usuario sin conocimientos técnicos. Estas preguntas están **distribuidas entre los distintos módulos funcionales del sistema** (Personas, Departamentos, Registros, Trámites y Servicios, Empleo Público, etc.) y cubren diferentes tipos de ayuda (significado, reglas, obligatoriedad y descripción de uso).

Además, se han clasificado en tres niveles de dificultad (fácil, medio y difícil) en función de la complejidad conceptual y técnica de la respuesta esperada.

La evaluación se ha realizado utilizando la versión final del prompt, diseñada específicamente para generar consultas SPARQL válidas y adaptadas al vocabulario ontológico del sistema. Este prompt, desarrollado mediante un proceso iterativo de mejora (descrita previamente en el apartado *III. Interfaz de lenguaje natural*), ha sido clave para guiar al modelo de lenguaje en la producción de respuestas precisas y ejecutables sobre datos reales.

Para cada pregunta, se ha evaluado manualmente la respuesta generada por el sistema según los siguientes criterios:

- *Exactitud*: se clasifica cada respuesta como correcta, parcialmente correcta o incorrecta, en función de si responde completamente a la intención de la pregunta, de forma parcial o con errores semánticos. Se valida con los datos de Oracle.
- *SPARQL generado*: se comprueba si la consulta generada es sintácticamente válida y compatible con el motor Ontop. Y mapeo de datos esperado.
- *Observaciones*: se anotan observaciones para registrar comportamientos, errores recurrentes o aspectos destacables.

A continuación se muestra el resultado de la evaluación sobre el conjunto de preguntas diseñadas:

Tabla 1. Resultados de la evaluación de la ayuda inteligente contextual.

Evaluación de la Ayuda Inteligente		
Métrica	Resultado	%
Respuesta correcta	55	60,4%
Respuesta parcial	6	6,6%
Respuesta incorrecta	30	33,0%

Tabla 2. Distribución de resultados por nivel de dificultad y tipo de respuesta.

Nivel	Correctas	Parcial	Incorrectas	Total	% Correc.	% Parcial	% Incorrec.
Fácil	33	2	8	43	76,7 %	4,7 %	18,6 %
Medio	13	2	7	22	59,1 %	9,1 %	31,8 %
Difícil	9	2	15	26	34,6 %	7,7 %	57,7 %

Los porcentajes se calculan respecto al total de preguntas por nivel.

Figura 3. Distribución porcentual de respuestas por nivel de dificultad: fácil, medio y difícil.

Fuente: elaboración propia.

Durante la evaluación, se identificaron diversas fortalezas y debilidades del sistema de ayuda inteligente contextual. Entre los **aspectos más destacables**, se observó una **alta precisión en preguntas orientadas al significado y a la descripción de uso de los campos**, especialmente en aquellos módulos con mayor representación en el vocabulario del prompt, como *Personas*, *Departamentos* y *Trámites y Servicios*. Asimismo, el sistema mostró un **buen rendimiento en niveles de dificultad fácil y medio**, donde el lenguaje natural utilizado coincidía en gran medida con los patrones previamente aprendidos a través de los ejemplos incluidos en el prompt. Otro aspecto positivo fue el tiempo medio de respuesta, situado en torno a 1,36 segundos, lo que lo convierte en una herramienta ágil y operativa en entornos interactivos. Cabe destacar que, al repetir una consulta previamente procesada, el sistema era capaz de devolver la respuesta en tan solo 0,002 segundos gracias al mecanismo de reutilización de contexto.

No obstante, **también se detectaron errores frecuentes**, principalmente en preguntas con estructuras más abiertas, ambigüedades semánticas o términos poco representados en el conjunto de entrenamiento del modelo. En algunos casos, las respuestas eran incompletas o carecían de contexto, debido a la omisión de filtros por módulo o a una interpretación parcial de la intención del usuario. También se produjeron **fallos en preguntas que requerían devolver múltiples propiedades**, como aquellas relacionadas con campos de ayuda en las que se esperaba, por ejemplo, tanto una definición como sus reglas asociadas.

En cuanto a las **limitaciones identificadas**, se constató que **el rendimiento del sistema depende** de forma crítica **de la calidad del prompt utilizado**. El modelo necesita instrucciones claras, vocabulario controlado y ejemplos adecuados para producir respuestas útiles y coherentes. Por ejemplo, para la pregunta "¿Qué información debo incluir en el campo de procedimiento de cobro si el trámite tiene tasas?", se requirió reformulación "fase fin" para obtener la respuesta esperada. Además, **el sistema no proporciona retroalimentación al usuario cuando no puede generar una consulta válida**, lo que puede dificultar su usabilidad en situaciones reales. Por ejemplo, para la pregunta "¿Qué pasa si no conozco el departamento al que pertenece el registro?" , no encuentra información relevante para el usuario. Por último, en algunos casos, **la ausencia de datos en la base Oracle** para ciertos campos **impidió validar las respuestas**, a pesar de que la consulta SPARQL estuviera correctamente formulada desde el punto de vista sintáctico. Es el caso de la pregunta, "¿Qué significa el campo tasas/pagos en la solicitud y cómo influye en la tramitación del procedimiento?" una vez validada e incluido el campo *tasas*, devolvió los datos esperados.

Reevaluación con un prompt mejorado y comparación de resultados.

Para comprobar el impacto del diseño del prompt en la calidad de las respuestas, se seleccionó un **subconjunto de 35 preguntas que fueron clasificadas como incorrectas o parcialmente correctas en la evaluación inicial**. Estas preguntas se volvieron a ejecutar utilizando una versión anterior del prompt (v1), menos refinada y con menor número de ejemplos representativos. El objetivo era comparar si las mejoras introducidas en la versión final del prompt (v2) —como el control estricto del vocabulario, instrucciones explícitas y aprendizaje por analogía (few-shot learning)— influían de forma cuantificable en la exactitud semántica de las respuestas.

El prompt v2 mantuvo la misma estructura general que el v1, pero con mejoras clave en cada uno de sus componentes:

1. El diccionario semántico fue ampliado y depurado, incorporando más propiedades y utilizando descripciones más claras y específicas del dominio ontológico.
2. Las instrucciones fueron reformuladas para ser más estrictas y directas, reduciendo ambigüedad y delimitando mejor el formato de la respuesta esperada**.**
3. Se añadieron más ejemplos (few-shot learning), incluyendo casos con mayor complejidad y variedad, lo que ayudó al modelo a generalizar mejor las consultas.

Esta **mejora** progresiva del **prompt permitió aumentar** significativamente la **precisión** de las **respuestas** generadas, **reducir errores** por ambigüedad e **incrementar** la **relevancia semántica**, como se analiza en el apartado de evaluación.

TABLE 3. COMPARATIVA DE RESULTADOS ENTRE VERSIONES DEL PROMPT.

Tipo de Pregunta	Total Preguntas	Correctas prompt v2	Mejora (%)
Significado	6	5	83
Reglas	6	6	100
Obligatoriedad	6	5	83,3
Relación entre campos	1	0	0
Descripción de uso	15	12	80
Ejemplo de uso	1	1	100
TOTAL GENERAL	35	28	80

La evaluación detallada del subconjunto de 35 preguntas que no fueron correctamente respondidas con el prompt original (v1) revela una mejora tras la aplicación del prompt optimizado (v2). Los resultados muestran que el rediseño del prompt permitió resolver correctamente 28 de las 35 preguntas previamente fallidas, lo que representa una mejora global del 80% en esta muestra.

La mejora observada en la generación de respuestas se debe principalmente a cuatro elementos introducidos en el

prompt v2: (1) la inclusión de ejemplos específicos por campo y módulo, que proporcionaron patrones concretos que el modelo pudo imitar con precisión; (2) la mejora en la identificación del tipo de pregunta, lo que permitió seleccionar la propiedad ontológica adecuada según si se pedía una definición, una regla, un ejemplo o una obligatoriedad; (3) el aumento del parámetro *max_tokens* a 300, que evitó respuestas truncadas y permitió completar consultas complejas; y (4) la reutilización de estructuras previamente resueltas, gracias al reconocimiento contextual que el modelo aplica para formular consultas similares a ejemplos ya exitosos. Estos factores han resultado decisivos para transformar respuestas incorrectas o incompletas en consultas SPARQL válidas, completas y ejecutables. Cabe señalar que seis preguntas no pudieron resolverse correctamente debido a limitaciones estructurales del sistema, como la ausencia de información en el archivo OBDA o la falta de implementación de ciertas relaciones entre campos en la ontología, lo cual marca una frontera clara entre las capacidades actuales del sistema y su posible evolución futura.

La comparación entre versiones del prompt confirma que el diseño y optimización de éste tiene un impacto directo en el rendimiento del sistema. Aunque el modelo de lenguaje utilizado es el mismo, su comportamiento varía significativamente según el contexto proporcionado en el prompt. La incorporación de ejemplos concretos, la delimitación del vocabulario permitido y la claridad en las instrucciones han contribuido a mejorar la precisión semántica y reducir errores de interpretación.

Esta evidencia valida el papel del *prompt engineering* como elemento clave en interfaces de lenguaje natural conectadas a sistemas semánticos, y refuerza la necesidad de abordar el diseño del prompt como una tarea crítica y estratégica dentro de este tipo de soluciones.

II. Evaluación de Detección semántica de errores

El sistema desarrollado incorpora una funcionalidad orientada a la detección automática de errores semánticos en los datos del dominio atención al ciudadano, representando los fallos identificados como instancias OWL de la clase *:Error* y sus subclases especializadas. Esta evaluación tiene como objetivo validar la viabilidad y efectividad de este mecanismo en un entorno simulado con datos ficticios, mediante la ejecución de consultas SPARQL sobre el motor Ontop conectado a Oracle a través de mapeos OBDA.

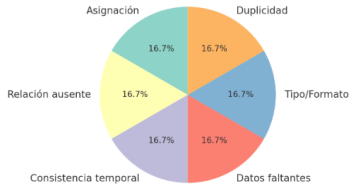
FIGURA 3. FRAGMENTO DE :ERRORCONSISTENCIA EN EL FICHERO DE MAPEO ODBA.

```
mappingId errorTramiteTipo
target :error_tramite_{TRS_CODIGO} a :ErrorConsistencia .
source SELECT TRS_CODIGO FROM SYSTEM.AC_TRAMITES_SERVICIOS WHERE TRS_DEPCOD IS NULL

mappingId errorTramiteDescripcion
target :error_tramite_{TRS_CODIGO} :errorDescripcion "El trámite no tiene departamento asignado" .
source SELECT TRS_CODIGO FROM SYSTEM.AC_TRAMITES_SERVICIOS WHERE TRS_DEPCOD IS NULL

mappingId errorTramiteFecha
target :error_tramite_{TRS_CODIGO} :errorFechaDetec "2025-05-01T00:00:00"^^xsd:dateTime .
source SELECT TRS_CODIGO FROM SYSTEM.AC_TRAMITES_SERVICIOS WHERE TRS_DEPCOD IS NULL

mappingId errorTramiteDetectadoEn
target :error_tramite_{TRS_CODIGO} :errorDetectadoEn :tramite_{TRS_CODIGO} .
source SELECT TRS_CODIGO FROM SYSTEM.AC_TRAMITES_SERVICIOS WHERE TRS_DEPCOD IS NULL
```

Las consultas fueron diseñadas para detectar errores típicos del dominio como:

1) *Error de asignación*: cuando una entidad no está asociada a otra obligatoria según la lógica del dominio.
2) *Error de relación ausente*: cuando falta una relación entre entidades que debería existir entre entidades.
3) *Error de consistencia temporal*: cuando las fechas registradas no siguen un orden lógico.
4) *Error de datos faltantes*: cuando campos esenciales u obligatorios están vacíos.
5) *Error de tipo o formato*: cuando un valor no cumple el formato esperado.
6) *Error de duplicidad*: cuando dos o más registros representan la misma entidad de manera redundante o inconsistente.

FIGURA 4. DISTRIBUCIÓN DE TIPOS DE ERRORES SEMÁNTICOS DETECTADOS.



Fuente: elaboración propia.

Cada consulta se evaluó manualmente según los siguientes criterios:

- *Relevancia*: se valora si el error detectado corresponde a una situación semánticamente incorrecta que el sistema debería evitar.
- *Exactitud*: Se comprueba que los resultados identifican efectivamente casos reales (o simulados) de error.
- *Tiempo de respuesta*: se mide el tiempo medio de ejecución de la consulta sobre el sistema.
- *Cobertura*: se analiza si la consulta abarca adecuadamente los casos esperados.
- *Trazabilidad semántica*: se verifica que cada error esté correctamente representado en la ontología como instancia de la clase *:Error* o subclases específicas (*:ErrorConsistencia*, *:ErrorDuplicidad*, *:ErrorAsignacion* etc.).

- *Reutilización*: se considera si la consulta puede integrarse en procesos futuros de validación automatizada.
- *Observaciones*: se documentan aspectos relevantes como el uso de funciones SPARQL especiales o subconsultas.

Este enfoque semántico permite detectar errores que podrían haber pasado desapercibidos en las validaciones tradicionales, especialmente en casos derivados de cargas masivas, migraciones de datos, modificaciones posteriores o cambios en las reglas de negocio.

Los resultados se resumen en la siguiente tabla, en la que se describe cada tipo de error, la entidad afectada y las propiedades evaluadas.

TABLA 4. RESULTADOS EVALUACIÓN - DETECCIÓN SEMÁNTICA DE ERRORES MEDIANTE CONSULTAS SPARQL.

Tipo error	Entidad afectada	Relevancia	Exactitud	Tiempo Respuesta (s)	Cobertura	Trazabilidad Semántica	Reutilización	Observaciones
Error de asignación	TramiteServicio	Sí	Trámite 101	0.126	Detecta trámites sin departamento	Sí	Sí	Implementado con 4 mapeos modulares
Error de relación ausente	Personas	Sí	Persona 34	0.051	Detecta personas sin unidad asignada	Sí	Sí	
Error de consistencia temporal	FaseInicio	Sí	Fase Inicio 20	0.214	Detecta fechas fin anteriores a inicio	Sí	Sí	
Error de datos faltantes	EmpleoPublico	Sí	Empleo público 51	0.62	Detecta campos nulos obligatorios	Sí	Sí	
Error de tipo o formato	Registros	Sí	Registro 301	0.015	Detecta campos numéricos mal formateados	Sí	Sí	Usa REGEXP_LIKE para validar formato numérico
Error de duplicidad	Personas	Sí	Personas 1, 201 y 202	0.511	Detecta duplicados por identificación	Sí	Sí	Se requiere subconsulta para contar duplicados

Los **resultados** obtenidos **reflejan** que **el sistema es capaz de identificar con éxito diversos tipos de errores semánticos** distribuidos en múltiples módulos funcionales. Cada error ha sido representado correctamente como una instancia OWL, lo que proporciona trazabilidad y capacidad de razonamiento sobre su origen y naturaleza.

Los **tiempos de respuesta** fueron todos inferiores a un segundo, con un **promedio de 0,26 s**, lo cual demuestra que este enfoque es eficiente y viable para validaciones automáticas en producción. Además, las consultas fueron diseñadas de forma reutilizable y extensible, lo que permitiría su integración futura en sistemas de control de calidad de datos o monitorización.

Sin embargo, existen algunas **limitaciones** importantes:

1) La evaluación se ha centrado en **seis tipos de errores**, seleccionados por su relevancia y viabilidad técnica, lo que **no garantiza una cobertura exhaustiva de todos los posibles fallos**.
2) La **ejecución** de las consultas se ha realizado **directamente desde Ontop de forma manual**.
3) Aún **no se ha implementado** un **mecanismo** de **notificación ni una interfaz** de visualización de errores.

Por tanto, el **sistema** actual debe considerarse como una **prueba de concepto funcional** con capacidad real de detección, pero con margen de evolución hacia un sistema más automatizado y completo.

III. Evaluación Consultas personalizadas en lenguaje natural

Para evaluar la funcionalidad de generación de consultas personalizadas en lenguaje natural, se diseñaron **14 preguntas representativas** de casos de uso en el dominio de atención ciudadana. Estas preguntas fueron formuladas de forma libre y procesadas por el sistema, que las traduce automáticamente a consultas SPARQL para su ejecución mediante Ontop sobre la base de datos Oracle. El objetivo fue analizar si el sistema era capaz de devolver resultado esperados generando consultas válidas en un tiempo aceptable.

Cada consulta fue clasificada por su nivel de dificultad (fácil, media o difícil) y se evaluó con base en tres métricas principales: el tiempo de respuesta, la exactitud semántica (es decir, si la consulta respondía correcta, incorrecta o parcialmente), y posibles observaciones relacionadas con la cobertura funcional o disponibilidad de datos.

Los resultados mostraron que **el sistema fue capaz de generar consultas SPARQL correctas para todas las preguntas evaluadas**. En total, 14 de 14 consultas fueron correctas **desde el punto de vista sintáctico y semántico**, lo que representa una tasa de éxito del 100%. Aunque, en **dos casos** (consultas ID 13 y 14), **la base de datos no contenía los valores esperados**, por lo que **no se devolvieron resultados**, aunque la consulta fue correctamente formulada. Este comportamiento se considera igualmente válido, ya que **el sistema actuó correctamente frente a la ausencia de datos**.

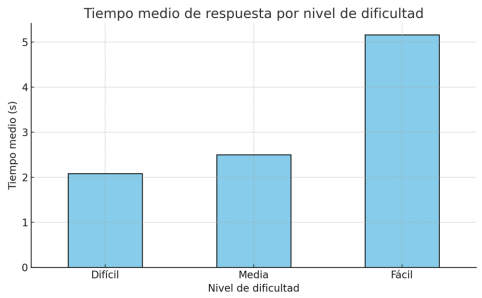
TABLA 5. RESULTADOS EVALUACIÓN - CONSULTAS PERSONALIZADAS EN LEGUAJE NATURAL.

Métrica	Resultado
Total de consultas evaluadas	14
Consultas con SPARQL correcto	14
Consultas que devolvieron datos	12
Consultas que no devolvieron datos	2
Tasa de exactitud semántica	100%
Tiempo medio de respuesta global (s)	2,70
Tiempo medio (Fácil) (s)	5,16
Tiempo medio (Media) (s)	2,5
Tiempo medio (Difícil) (S)	2,08

El tiempo medio de respuesta global fue de 2,70 segundos, con una ligera variación según el nivel de dificultad. La consulta más lenta (8,89 segundos) fue la primera, posiblemente debido al mayor volumen de

resultados retornados (todas las personas con DNI). A continuación, se muestra un gráfico comparativo del tiempo medio de respuesta por nivel de dificultad, que pone de manifiesto que **no existe una correlación directa entre la complejidad semántica de la pregunta y el tiempo de ejecución**, lo cual es un indicio de **buena optimización del sistema**:

FIGURA 5. TIEMPO MEDIO DE RESPUESTA POR NIVEL DE DIFICULTAD.



Fuente: elaboración propia.

El sistema de consultas personalizadas demuestra ser eficaz y funcional. La integración del modelo semántico con una arquitectura Ontop + Oracle ha permitido generar respuestas exactas a preguntas en lenguaje natural, incluso en escenarios complejos. Estos resultados validan el uso de tecnologías semánticas para mejorar la accesibilidad y la interacción con sistemas administrativos, permitiendo que usuarios no técnicos consulten los datos sin necesidad de conocer el lenguaje SPARQL.

Sin embargo, es **importante tener en cuenta que el lenguaje natural presenta una gran ambigüedad**, y la casuística es prácticamente infinita, **por lo que no puede garantizarse que el 100% de las consultas personalizadas se resuelvan correctamente**. Además, el ***dataset*** utilizado en esta evaluación estaba **limitado** a poco más de un centenar de registros, lo cual dista mucho de la complejidad y volumen de datos que tendría un sistema real con millones de entradas. Esto **condiciona los resultados y refuerza la necesidad de futuras pruebas en entornos más amplios y realistas**.

REPOSITORIO DEL PROYECTO

Toda la información relacionada con este trabajo, incluyendo la implementación del código, la documentación técnica extendida, los mapeos OBDA, los prompts utilizados (versión 1 y 2), así como los resultados completos de la evaluación, se encuentra disponible en el siguiente **repositorio de GitHub**:

https://github.com/amvajua/TFM-Ayuda-Inteligente

Este material puede servir como referencia complementaria para quienes deseen consultar detalles técnicos, reproducir el prototipo o explorar futuras ampliaciones del sistema.

CONCLUSIONES

El **objetivo** de este Trabajo de Fin de Máster ha sido **estudiar la viabilidad de aplicar inteligencia en la web** a un sistema de gestión **del ámbito de atención al ciudadano**, **diseñar** una **solución funcional** basada en ese enfoque y **evaluar su efectividad en un entorno simulado**. La propuesta parte de una situación real en la que se detectan **mejoras aplicando inteligencia en la web** en la ayuda al usuario, las consultas disponibles y el control de errores, lo que motivó el diseño de una solución que abordase estos tres puntos clave: ofrecer una **ayuda más útil y contextual**, permitir **consultas personalizadas en lenguaje natural** y **detectar errores antes de que afecten al usuario**.

Una de las **principales ventajas** de la solución propuesta es que **no** ha sido **necesario modificar la arquitectura ni** el **modelo de datos relacional de la aplicación original**. En su lugar, se ha trabajado en paralelo, construyendo una ontología que representa el conocimiento del dominio y utilizando técnicas de inteligencia artificial para facilitar la interacción del usuario con el sistema. Esto permite preservar la estructura existente y, al mismo tiempo, dotar al sistema de una capa de inteligencia y flexibilidad adicional.

Además, la **arquitectura desarrollada** es fácilmente **reutilizable en otros dominios**, así como su mantenimiento. Basta con cambiar la ontología, adaptar los mapeos a la nueva base de datos y ajustar el prompt del modelo de lenguaje a las nuevas entidades y relaciones. Esta reutilización es posible gracias a que la ontología ha sido diseñada de forma modular, con clases bien definidas, propiedades claras y relaciones explícitas entre entidades del dominio. Esta estructura facilita la extensión o sustitución de conceptos sin comprometer la coherencia del modelo.

La **ayuda inteligente y** el sistema de **consultas personalizadas se integran mediante un botón sencillo** en *React* y *Spring Boot*, lo que permite su incorporación en cualquier aplicación sin grandes complicaciones.

En cuanto a la **detección semántica de errores**, aunque su alcance aún es limitado, se ha **demostrado su viabilidad conceptual y técnica**. Es una línea con un alto potencial de desarrollo futuro, especialmente útil para reforzar la calidad y coherencia de los datos en entornos complejos. Su valor radica en poder anticipar problemas antes de que afecten directamente al usuario final.

En la **evaluación realizada**, se analizaron por separado las funcionalidades de **ayuda inteligente**, consultas personalizadas y detección de errores. En el caso de la ayuda inteligente, se realizó una primera prueba con un

prompt base que permitió obtener respuestas relevantes en la mayoría de los casos. A partir de los casos no resueltos, se diseñó una segunda prueba con un prompt mejorado, centrado en dar más contexto y ejemplos. Esta segunda evaluación permitió recuperar varias respuestas inicialmente incorrectas o incompletas, lo que evidenció que **la calidad del prompt influye directamente en la precisión semántica del sistema**. En general, el sistema respondió con mayor claridad, reduciendo la ambigüedad y adaptando mejor la respuesta al contexto funcional del usuario. En cuanto a las **consultas en lenguaje natural**, se diseñaron preguntas representativas de casos ficticios y se evaluó la capacidad del sistema para generar consultas SPARQL válidas y obtener resultados correctos en base a los datos de Oracle. La mayoría fueron interpretadas adecuadamente, aunque algunas presentaron limitaciones debido a restricciones del mapeo OBDA o a la falta de cobertura en la ontología. Estos resultados indican que, si bien **la solución es efectiva**, **requiere ajustes continuos para mantener su precisión a medida que evoluciona el dominio**. Y la complejidad de las consultas dependerá de lo que el usuario solicite en lenguaje natural desde del frontend. En cuanto al **tiempo de respuesta** de las mismas, se vio que **no existe una correlación directa** entre la **complejidad semántica de la pregunta y el tiempo de ejecución**. Por último, se ha incorporado una funcionalidad de **detección semántica de errores** que, aunque más acotada en su alcance, ha demostrado ser **útil para identificar incoherencias conceptuales en los datos a través de consultas SPARQL**. Esta capacidad complementa el sistema al contribuir a la mejora de la calidad de la información, incluso sin procesos de detección completamente automatizados, lo que refuerza el valor del enfoque semántico adoptado.

Ahora bien, este trabajo presenta también algunas **limitaciones importantes**. Por un lado**,** la **evaluación** se ha realizado en un entorno simulado, con datos de prueba y **sin participación** de **usuarios reales y dataset acotado**, lo que **limita** la **generalización de los resultados**. Por otro lado, aunque el uso de modelos de lenguaje ha sido efectivo, **su rendimiento depende en gran medida del diseño del prompt y del contexto**, lo que puede requerir ajustes continuos si cambia el dominio o los datos. Además, **la detección de errores no está automatizada y requiere aún intervención manual**, lo que reduce su impacto real en entornos productivos. A ello se suma el **coste asociado al uso de modelos LLM externos**, especialmente si se integran en sistemas con alta demanda de consultas; como GPT-3.5 o GPT-4, cuyo coste por consulta puede oscilar entre 0,0006 € y 0,0035 € dependiendo del modelo y la longitud del mensaje. Si bien el impacto es bajo para sistemas con pocas consultas, puede incrementarse significativamente en entornos con alta demanda, lo que requiere una evaluación cuidadosa de escalabilidad y sostenibilidad económica. Como alternativa a los modelos externos, **la administración pública podría optar por desplegar modelos de lenguaje en su propia infraestructura**, utilizando soluciones de código abierto como *LLaMA* o *Mistral*. Esta estrategia permitiría mantener el control sobre los datos, eliminar costes por uso y adaptar el comportamiento del modelo a las necesidades específicas del entorno público. También es necesario considerar la **complejidad de mantenimiento del sistema**, ya que involucra múltiples componentes (ontología, mapeos, backend, frontend y API externa) que deben mantenerse alineados. Por último, aunque el sistema es escalable desde el punto de vista técnico, su **despliegue en un entorno real requeriría** una fase adicional de **adaptación**, **validación** con **usuarios reales** y **coordinación** con los **responsables del sistema actual**.

En definitiva, los resultados obtenidos confirman que **sí vale la pena aplicar inteligencia en la web al sistema**, ya que **mejora la eficiencia**, **accesibilidad** y **calidad del servicio**. Ahora bien, es importante tener en cuenta que **su implementación** conlleva un **proceso de maduración y** una **complejidad comparable a la del propio sistema al que se quiere adaptar**.

La arquitectura propuesta, basada en ontologías, consultas SPARQL y el uso de modelos de lenguaje, ha demostrado ser efectiva, adaptable y con potencial de reutilización en otros contextos similares. A pesar de las limitaciones identificadas y los retos pendientes para su despliegue en un entorno real, se concluye que la **integración de inteligencia semántica e IA aporta un valor añadido** claro y tangible a los sistemas de atención al ciudadano.

I. Trabajos futuros

Aunque este trabajo se ha basado en un prototipo ficticio y simplificado, ha servido como punto de partida para comprobar que aplicar inteligencia en la web puede mejorar la interacción con el sistema y aportar funcionalidades útiles. A partir de aquí, hay varias ideas que podrían explorarse si se quisiera seguir desarrollando esta propuesta.

Por un lado, sería interesante **mejorar la comprensión del lenguaje natural y costes**. Entrenando **un modelo LLM local adaptado al dominio y ajustando el prompt;** podría ayudar a interpretar consultas más complejas y afinar aún más las respuestas.

También se podría **automatizar la detección de errores**. Ahora mismo se hace manualmente con consultas SPARQL, pero sería muy útil que el sistema pudiera lanzar esas validaciones de forma periódica, detectar incoherencias y avisar automáticamente si algo falla.

Otra mejora posible sería **ampliar las opciones de exportación**. De momento se pueden generar listados en Excel, pero se podrían ofrecer otros formatos como PDF, CSV o integrarlo con sistemas institucionales de informes.

Un paso importante sería **validar con usuarios reales**. Aunque el prototipo ha demostrado ser funcional, sería necesario probarlo con personas que realmente usan estos sistemas para saber si les resulta útil, fácil de usar y si resuelve sus necesidades.

Otro aspecto a tener en cuenta es **definir una estrategia de mantenimiento y evolución del sistema**. Como se trata de una solución compuesta por varios elementos interconectados (ontología, mapeos, backend, frontend y modelo de lenguaje), sería necesario contar con una forma de mantenerla alineada y actualizada a lo largo del tiempo, incluyendo el mantenimiento de la documentación.

También sería interesante **incorporar soporte multilingüe**, permitiendo que el sistema funcione en valenciano (cooficial en la Comunidad Valenciana) y en inglés. Esto podría resolverse adaptando el prompt del modelo de lenguaje a cada idioma y utilizando bibliotecas o servicios de traducción automática en el frontend, sin necesidad de modificar el núcleo de la arquitectura. A nivel ontológico, bastaría con incluir etiquetas rdfs:label en los distintos idiomas para cada clase y propiedad, por ejemplo: rdfs:label "Departamento"@es, rdfs:label "Departament"@va, rdfs:label "Department"@en. De esta manera, el sistema puede identificar los conceptos semánticos independientemente del idioma del usuario, manteniendo una única estructura lógica centralizada.

Y por último, una línea futura especialmente relevante sería la **adaptación del sistema para personas con discapacidad**. Esto podría abordarse desde el propio frontend, aplicando buenas prácticas de accesibilidad web (compatibilidad con lectores de pantalla, navegación por teclado, contraste adecuado, etc.) sin necesidad de modificar la arquitectura base del sistema.

Referencias

[1] S. Firmenich, "Interfaz conversacional para la recolección de datos en la Web Semántica," Tesina de grado, Univ. Nacional de La Plata, 2023.

[2] D. M. Arteaga Meléndez, "Diseño de un modelo explicativo basado en ontologías aplicado a un chatbot conversacional," Tesis de licenciatura, Pontificia Univ. Católica del Perú, 2023.

[3] M. Ciancio, "Creación de un prototipo de chatbot que permita interactuar con la historia del Ecuador registrada en periódicos antiguos," Tesis de grado, Univ. de Cuenca, 2024.

[4] E. J. Huaranga Junco, "Distilling ontologies from large language models," Trabajo Fin de Máster, Univ. Politécnica de Madrid, 2023.

[5] J. Gómez de Agüero Muñoz, "MethoOntoChat: Asistente conversacional del proceso metodológico de creación de ontologías basado en modelos del lenguaje," Trabajo Fin de Máster, Univ. Politécnica de Madrid, 2024.

[6] J. Tupayachi, H. Xu, O. A. Omitaomu, M. C. Camur, A. Sharmin, and X. Li, "Towards next-generation urban decision support systems through AI-powered construction of scientific ontology using large language models—A case in optimizing intermodal freight transportation," *Smart Cities*, vol. 7, no. 5, pp. 2392–2421, 2024.

[7] M. Paneque Romero, "Explotación de la semántica de dominio orientada a la integración, estandarización y análisis de datos," Tesis doctoral, Univ. de Málaga, 2022.
.

[8] K. A. Lagos Ortiz, "Sistema de ayuda a la decisión basado en ontologías para el diagnóstico y prevención de las enfermedades en cultivos," Tesis doctoral, Univ. de Murcia, 2020.

[9] A. Martínez, "Revisión de los chatbots basados en inteligencia artificial en la administración pública: Hacia una arquitectura para el gobierno," *Espacios Públicos*, vol. 25, no. 1, pp. 45–60, 2022.

[10] R. J. Celi-Párraga, E. A. Varela-Tapia, I. L. Acosta-Guzmán, and N. R. Montaño-Pulzara, "Técnicas de procesamiento de lenguaje natural en la inteligencia artificial conversacional textual," *AlfaPublicaciones*, vol. 3, no. 4.1, pp. 40–52, 2021.

[11] U. Lopez, E. Cabrio, A. Freitas, and S. Walter, "Evaluating Question Answering over Linked Data: Challenges and Open Issues," *Semantic Web*, vol. 8, no. 6, pp. 1045–1070, 2017.

[12] B. J. Brito Karadjian, "Sistema de generación de consultas SQL basado en peticiones de lenguaje natural a partir de un chatbot," Trabajo Fin de Grado, Univ. Europea de Valencia, 2024.

[13] Suárez-Figueroa, M. C., & Gómez-Pérez, A. (2012), "NeOn Methodology for Building Ontology Networks: A Scenario-based Approach. In M. C. Suárez-Figueroa, A. Gómez-Pérez, E. Motta, & A. Gangemi (Eds.)," Ontology Engineering in a Networked World (pp. 9–34). Springer. https://doi.org/10.1007/978-3-642-24794-1_2
.